to put variable in string use f
we can use + also

List Comprehensions to Create

Create file Comprehension.py

Suppose we want to make list that is divisible by 3

```
ls = []
for i in range(100):
    if i%3==0:
        ls.append(i)
print(ls)
```

o/p: [0, 3, 6, ... 99]

By List Comprehension, we can do this in one or two

```
ls = [i for i in range(100)] if i%3==0]
print(ls)
```

Dict list Comprehensions

```
dict1 = {i: f"item{i}" for i in range(10000)}
print(dict1)
dict1 = {i: f"item{i}" for i in range(1,10001) if i%
print(dict1)
```

or (1, 1000
means 2
1
stat

o/p: { 0 : i

To Reverse a dictionary: Key : Value → Value : Key

```
dict1 = {i: f"Item{i}" for i in range(5)}
dict21 = {value: key for key, value in dict1.items()}
print(dict1)
```

```
dict1 = {i: f"Item {i}" for i in range(5)}
dict2 = {value: key for key, value in dict1.items()}
print(dict1, dict2)
```

Set Comprehension :

```
dresses = {dress for dress in ["dress1", "dress2", "dress1",
                               "dress2", "dress1", "dress2"]}
print(dresses)
```

o/p: {'dress2', 'dress1'}

```
print(type(dresses))
```

o/p: Set

Generator :

```
evens = (i for i in range(100) if i%2 == 0)
print(type(evens))
```

o/p : <class 'generator'>

```
print(evens.__next__())
```

or

```
for item in events:
    print(items)
```

Generator comprehension

List Comprehension

List Comprehension represents creation of new list from an iterable object that satisfy a given condition.

Syntax:

new_list = [expression for item in iterable_object if statement]

There can be zero or more if statements

There can be one or multiple for loops

Ex:

```
list1 = [i+1 for i in range(20)]
list2 = [i for i in range(20) if i%2==0]
list3 = [i for i in range(11) if i%2==0 if i%3==
```

Q1: We have to add +1 in each item of list

```
list1 = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
new_list1 = []
for i in list1:
    new_list1.append(i+1)
print(new_list1)
```

Without comp
ensi

O/P: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

With list Comprehension

1) list2 = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

```
new_list2 = [i+1 for i in list2]
print (new_list2)
```

2)

```
new_list2 = [i+1 for i in range (20)]
print (new_list2)
```

Without list Comprehension

```
list1 = []
for i in range (20):
    if (i%2 == 0):
        list1.append(i)
print (list1)
```

With list Comprehension

```
list2 = [i for i in range (20)
        if i%2 == 0
print ("New list:", list2)
```

We can add more than one if statements.

if i%3==0, if i%

ol list Comprehensions :

```
matrix = [ ]
for i in range (3):
    matrix.append([])
for j in range (5):
    matrix[i].append(j)
print(matrix)
```

Without Comprehension

, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4]]

Nested List Comprehension

```
matrix = [[j for j in range(5)] for i in range(3)]
print(matrix)
```

fun are objects in Python object ... id ...

List Comprehension and lambda

Lambda: It is one liner function also called anonymous fu

Ex:

```
def add(a,b)
    return (a+b)
```

or

```
add = lambda x, y : x+
print(add(2,3))
```

```
def minus(x, y):
    return x-y
result = minus(3-4)
print(minus(9,4))
```

or

```
minus = lambda x,y
print(minus(9,4))
```

```
def square(a):
    return a*a
result = square(5)
print(result)
```